ĐẠI HỌC QUỐC GIA TP HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH

LAB: 6

QUẢN LÝ BỘ NHỚ

Lớp thực hành môn: HỆ ĐIỀU HÀNH

– **IT007.M11.CLC.1**

Sinh viên thực hiện:

MSSV: 20520322

Họ và tên: Nguyễn Thị Mỹ Trân

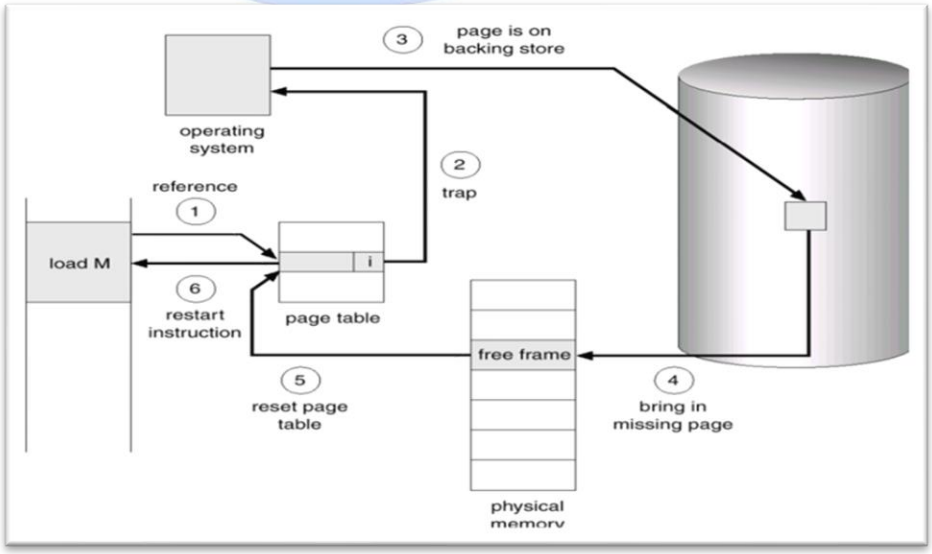
6.3.3 Câu hỏi chuẩn bị

1. Lỗi trang là gì? Khi nào xảy ra trường hợp lỗi trang? Vẽ lưu đồ mô tả cách hệ điều hành xử lý lỗi trang? Trình bày cách cài đặt Demand paging?

- **Định nghĩa lỗi trang:** Khi dò tìm các bảng trang để lấy thông tin cần thiết trong việc chuyển đổi địa chỉ, nếu nhận thấy trang được yêu cầu truy xuất hoặc trang được đánh dấu bất hợp lệ sẽ được gọi là lỗi trang.

- **Lỗi trang** xảy ra khi người dùng truy cập tới một trang không hợp lệ (nghĩa là trang không ở trong không gian địa chỉ của quá trình – không đang ở trong bộ nhớ chính) hoặc ta đang truy cập tới một trang hợp lệ nhưng đang ở bộ nhớ phụ (**swap space**).

- **Lưu đồ xử lý lỗi trang**



- Cách cài đặt Demand paging

Demand paging: Các trang của quá trình chỉ được nạp vào bộ nhớ chính khi được yêu cầu.

■ Khi có một tham chiếu đến một trang mà không có trong bộ nhớ chính (valid bit) thì phần cứng sẽ gây ra một ngắt (gọi là page-fault trap) kích khởi page-fault service routine (PFSR) của hệ điều hành.

■ PFSR:

- Chuyển process về trạng thái blocked
- Phát ra một yêu cầu đọc đĩa để nạp trang được tham chiếu vào một frame trống; trong khi đợi I/O, một process khác được cấp CPU để thực thi
- Sau khi I/O hoàn tất, đĩa gây ra một ngắt đến hệ điều hành; PFSR cập nhật page table và chuyển process về trạng thái ready

■ Bước 2 của PFSR giả sử phải thay trang vì không tìm được frame trống, PFSR được bổ sung như sau:

- Xác định vị trí trên đĩa của trang đang cần
- Tìm một frame trống:
  - Nếu có frame trống thì dùng nó
  - Nếu không có frame trống thì dùng một giải thuật thay trang để chọn một trang hy sinh (victim page)

- Ghi victim page lên đĩa; cập nhật page table và frame table tương ứng

- Đọc trang đang cần vào frame trống (đã có được từ bước 2); cập nhật page table và frame table tương ứng.

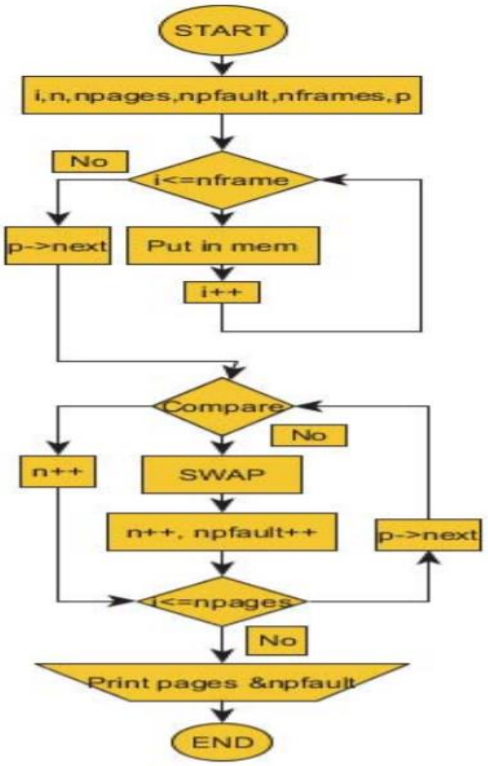
2. Tại sao phải thực hiện chiến lược thay thế trang?

Khi có lỗi trang xuất hiện, thì hệ thống phải chọn một trang nạn nhân để loại khỏi bộ nhớ, nhường không gian cho trang đang được yêu cầu. Nếu trang được chọn đã có sửa đổi thì hệ thống phải cập nhật lại nội dung trang này trên bộ nhớ phụ. Trang mới lấy từ bộ nhớ phụ chỉ việc chép chồng lên trang nạn nhân. Trang nạn nhân phải thường là trang phải là trang ít được khi sử dụng đến nhất. Vì nếu hệ thống chọn một trang được sử dụng rất nhiều làm trang nạn nhân, thì tổng chi phí cho việc thay trang của hệ thống tăng lên đáng kể. Vì vậy cần thực hiện chiến lược thay thế trang

3. Vẽ lưu đồ thuật toán mô tả cách thức xử lý của 3 giải thuật thay thế trang: FIFO, OPT, LRU? Vẽ sơ đồ trình bày thay thế trang bằng 3 giải thuật trên với chuỗi tham chiếu: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Giải sử có 3 khung trang và các khung trang ban đầu là rỗng. Xác định số Page Fault từ đó đưa ra đánh giá các giải thuật.

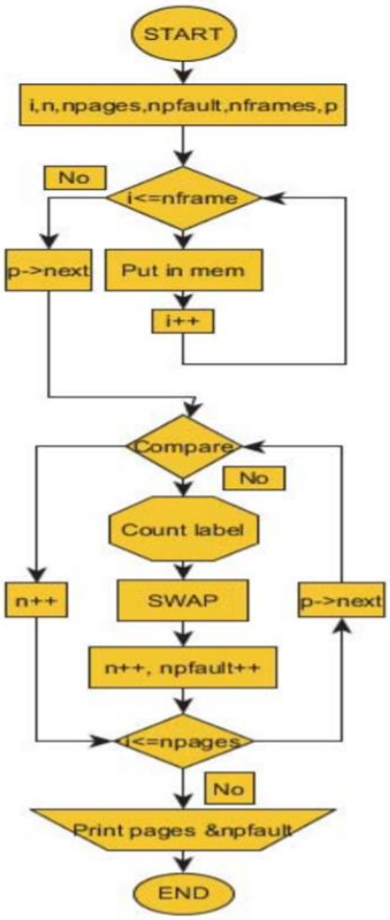
- **Giải thuật FIFO**



FIFO: 9 lỗi trang

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

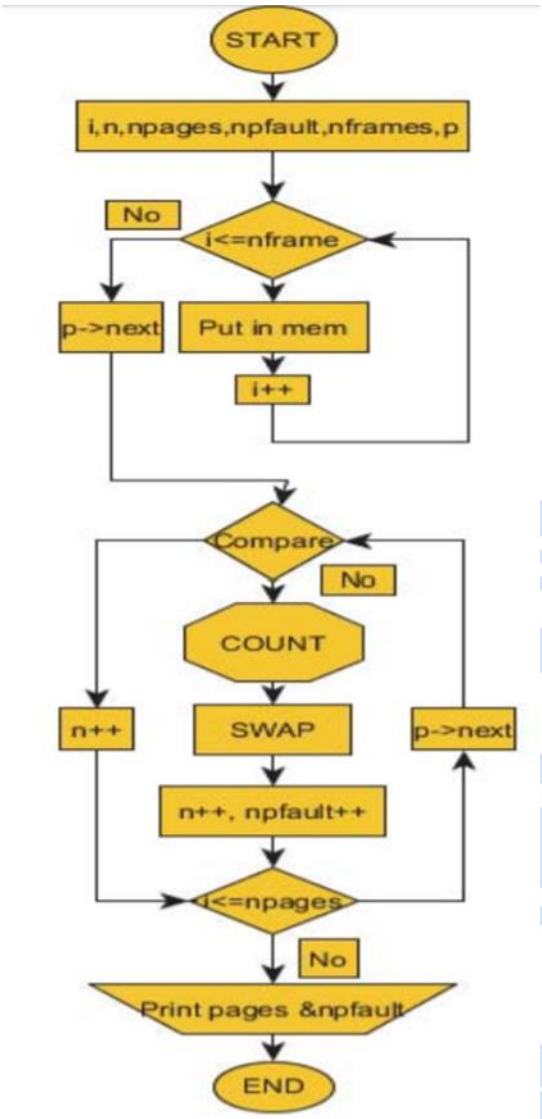
- Giải thuật OPT



OPT: 7 lỗi trang

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	0	0	0	0	0	3	3	2	2
	2	2	6	4	4	4	4	4	4	4	4
		1	1	1	1	1	1	1	1	1	1
*	*	*	*	*				*		*	

- Giải thuật LRU



LRU: 9 lỗi trang

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

6.4 Hướng dẫn thực hành

Source code 3 thuật toán tại file:

NguyenThiMyTran_20520322_LAB6.c

```
#include<stdio.h>

int OTP(int a[50], int frames[5], int pos, int frame, int n)

{


    int flag[5] = { 0,0,0,0,0 };

    int i = pos + 1;
```

```
for (i; i < n; i++)
{
	int t = 0;
	for (t = 0; t < frame; t++)

		if (frames[t] == a[i])
			flag[t] = 1;

	int count = 0;
	for (t = 0; t < frame; t++)
		if (flag[t] == 1)
			count++;

	if (count == frame)
	{
		int f = 0;
		for (f; f < frame; f++)
			if (frames[f] == a[i])
				return f;
	}
	if (i == n - 1)
	{
		int i = 0;
		for (i; i < frame; i++)
			if (flag[i] == 0)
				return i;
	}


}
```

```
}
int IsExist(int a, int temp[50], int index)
{
	int i = 0;
	for (i; i < index; i++)
		if (a == temp[i])
			return 1; //true 1
	return 0;//false 0
}

int LRU(int a[50], int frames[5], int pos, int frame, int n)
{
	int temp[50];
	int j = 0;
	int i = pos - 1;
	for (i; i >= 0; i--)
	{
		if (j == 0)
		{
			temp[j] = a[i];
			j++;
		}
		else
		{
			if (IsExist(a[i], temp, j) == 0)
			{
				temp[j] = a[i];
				j++;
			}
			if (j == frame)
```

```
                {
                        int t = 0;
                        for (t; t < frame; t++)
                                if (frames[t] == a[i])
                                        return t;
                }


        }
    }
}
int ChuoiMacDinh(int a[50])
{
    int n;
    n = 20;
    int b[20] = { 1,7,5,2,0,8,6,7,0,0,7 };
    int i = 0;
    for (i; i < n; i++)
        a[i] = b[i];
    return n;
}


int ChuoiThamChieuNhapBangTay(int a[50])
{
    int n;
    printf(" \nNhập số phần tử chuỗi tham chiếu: \n");
    scanf("%d", &n);
    printf(" \nNhập vào chuỗi tham chiếu: \n");
    int i = 0;
    for (i; i < n; i++)
        scanf("%d", &a[i]);
```

```
	return n;


}
int main()
{
	int i, j, n, a[50], frames[5], frame, k, available, count = 0;



	int input;
	printf("---- - Page Replacement algorithm---- -\n");
	printf("1. Chuỗi tham chiếu mặc định.\n");
	printf("2. Nhập chuỗi tham chiếu bằng tay.\n");
	scanf("%d", &input);

	if (input == 1)
		n = ChuoiMacDinh(a);
	if (input == 2)
		n = ChuoiThamChieuNhapBangTay(a);

	printf("\nNhập vào số khung trang :\n");
	scanf("%d", &frame);
	int y = 0;
	for (y; y < frame; y++)
		frames[y] = -1; // Giả sử ban đầu các frame trống
	printf("------Page Replacement algorithm-----\n");
	printf("1. Giải thuật FIFO\n");
	printf("2. Giải thuật OPT(optimal)\n");
	printf("3. Giải thuật LRU\n");
	int choose;
	scanf("%d", &choose);
```

```
if (choose == 1)
     printf("---FIFO Page Replacement algorithm---\n");
if (choose == 2)

     printf("------OTP Page Replacement algorithm-----\n");
if (choose == 3)
     printf("------LRU Page Replacement algorithm-----\n");



j = 0;
printf("\t|Chuỗi|\t|Khung trang");
for (k = 0; k < frame - 1; k++)
     printf("\t");
printf("|\n");


for (i = 0; i < n; i++)
{
     printf("\t|  %d  |\t", a[i]);
     available = 0; // trang không có sẵn
     for (k = 0; k < frame; k++)
          if (frames[k] == a[i]) // kiểm tra trang có sẵn
               available = 1; // trang có sẵn
     if (available == 0) // thay thế trang nếu không có sẵn
     {
          if (choose == 1)
          {
               frames[j] = a[i];
               j = (j + 1) % frame;


          }
          else if (choose == 2)
```

```
        {
            if (i < frame)
            {
                frames[j] = a[i];
                j++;
            }
            else
                frames[OTP(a, frames, i, frame, n)] = a[i];
        }
        else if (choose == 3)
        {
            if (i < frame)
            {
                frames[j] = a[i];
                j++;
            }
            else
                frames[LRU(a, frames, i, frame, n)] = a[i];
        }


        count++;
        printf("|");
        for (k = 0; k < frame; k++)
            printf("%d\t", frames[k]);
        printf("| F"); // Dấu hiệu nhận biết xảy ra lỗi trang
    }
    else
    {
        printf("|");
```

```
                for (k = 0; k < frame; k++)
                     printf("%d\t", frames[k]);
                printf("|");
            }
            printf("\n");
        }
        printf("Số trang lỗi là: %d\n", count);

        return 0;
}
```

Kết quả chạy thuật toán: Chuỗi vào có thể là ngẫu nhiên hoặc tự nhập. Đưa vào một chuỗi ngẫu nhiên và được kết quả như sau:

- **Giải thuật FIFO**

```
mytran@mytran-VirtualBox:~$ gcc NguyenThiMyTran_20520322_LAB6.c -o NguyenThiMyTran_20520322_LAB6
mytran@mytran-VirtualBox:~$ ./NguyenThiMyTran_20520322_LAB6
---- - Page Replacement algorithm---- -
1. Chuỗi tham chiếu mặc định.
2. Nhập chuỗi tham chiếu bằng tay.
1

Nhập vào số khung trang :
4
------Page Replacement algorithm-----
1. Giải thuật FIFO
2. Giải thuật OPT(optimal)
3. Giải thuật LRU
1
---FIFO Page Replacement algorithm---
        |Chuỗi|  |Khung trang                    |
        |  1  |  |1      -1      -1      -1      | F
        |  7  |  |1      7       -1      -1      | F
        |  5  |  |1      7       5       -1      | F
        |  2  |  |1      7       5       2       | F
        |  0  |  |0      7       5       2       | F
        |  8  |  |0      8       5       2       | F
        |  6  |  |0      8       6       2       | F
        |  7  |  |0      8       6       7       | F
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  7  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
        |  0  |  |0      8       6       7       |
Số trang lỗi là: 8
mytran@mytran-VirtualBox:~$
```

- **Giải thuật OPT**

```
mytran@mytran-VirtualBox:~$ ./NguyenThiMyTran_20520322_LAB6
---- - Page Replacement algorithm---- -
1. Chuỗi tham chiếu mặc định.
2. Nhập chuỗi tham chiếu bằng tay.
1

Nhập vào số khung trang :
4
------Page Replacement algorithm-----
1. Giải thuật FIFO
2. Giải thuật OPT(optimal)
3. Giải thuật LRU
2
------OTP Page Replacement algorithm-----
        |Chuỗi| |Khung trang                    |
        |  1  | |1      -1      -1      -1      | F
        |  7  | |1      7       -1      -1      | F
        |  5  | |1      7       5       -1      | F
        |  2  | |1      7       5       2       | F
        |  0  | |0      7       5       2       | F
        |  8  | |0      7       8       2       | F
        |  6  | |0      7       6       2       | F
        |  7  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  7  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
        |  0  | |0      7       6       2       |
Số trang lỗi là: 7
mytran@mytran-VirtualBox:~$
```

- **Giải thuật LRU**

```
mytran@mytran-VirtualBox:~$ ./NguyenThiMyTran_20520322_LAB6
---- - Page Replacement algorithm---- -
1. Chuỗi tham chiếu mặc định.
2. Nhập chuỗi tham chiếu bằng tay.
1

Nhập vào số khung trang :
4
------Page Replacement algorithm-----
1. Giải thuật FIFO
2. Giải thuật OPT(optimal)
3. Giải thuật LRU
3
------LRU Page Replacement algorithm-----
        |Chuỗi| |Khung trang                    |
        |  1  | |1      -1      -1      -1      | F
        |  7  | |1      7       -1      -1      | F
        |  5  | |1      7       5       -1      | F
        |  2  | |1      7       5       2       | F
        |  0  | |0      7       5       2       | F
        |  8  | |0      8       5       2       | F
        |  6  | |0      8       6       2       | F
        |  7  | |0      8       6       7       | F
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  7  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
        |  0  | |0      8       6       7       |
Số trang lỗi là: 8
mytran@mytran-VirtualBox:~$
```

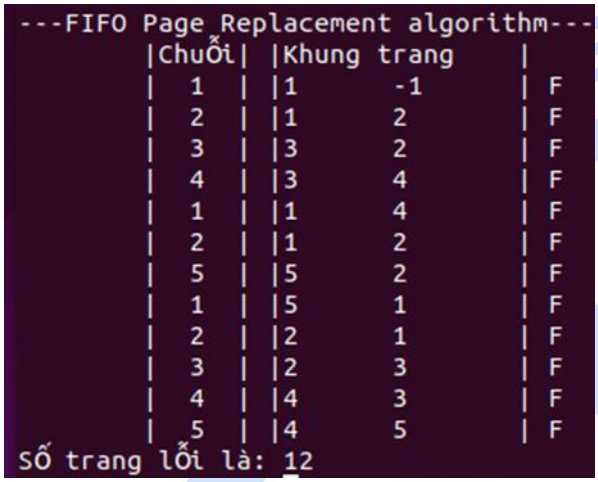
6.5 Bài tập ôn tập

1.Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

- Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng. Hiện tượng này gọi là nghịch lý Belady.

- Ví dụ:

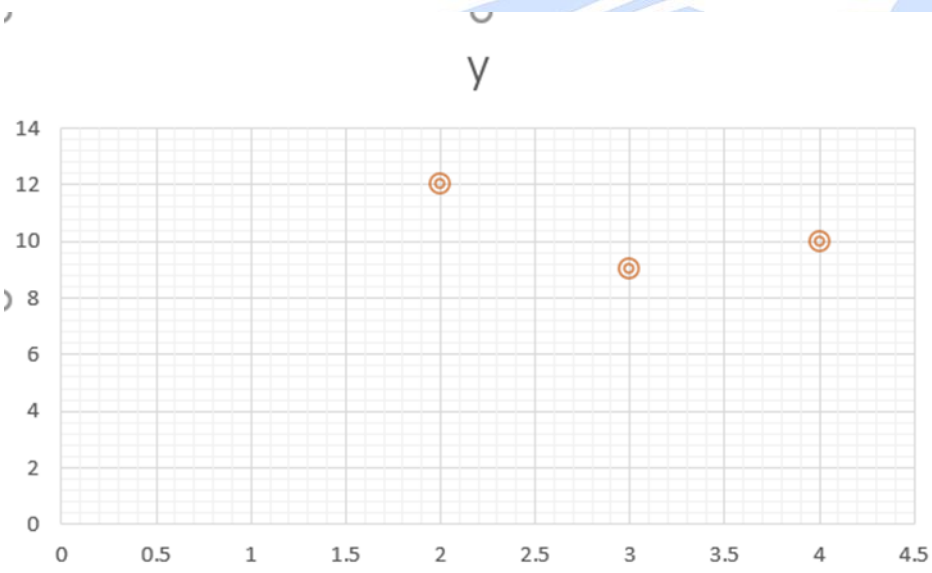
+ 2 khung trang



+ 3 khung trang

+ 4 khung trang

```
---FIFO Page Replacement algorithm---
        |Chuỗi| |Khung trang                    |
        |  1  | |1      -1      -1      -1      | F
        |  2  | |1      2       -1      -1      | F
        |  3  | |1      2       3       -1      | F
        |  4  | |1      2       3       4       | F
        |  1  | |1      2       3       4       |
        |  2  | |1      2       3       4       |
        |  5  | |5      2       3       4       | F
        |  1  | |5      1       3       4       | F
        |  2  | |5      1       2       4       | F
        |  3  | |5      1       2       3       | F
        |  4  | |4      1       2       3       | F
        |  5  | |4      5       2       3       | F
Số trang lỗi là: 10
```



Hình : Đồ thị biểu diễn nghịch lý Belady

2.Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.

❖ Giải thuật nào là bất khả thi nhất? Vì sao?

❖ Giải thuật nào là phức tạp nhất? Vì sao?

Vấn đề chính khi thay thế trang là chọn lựa một trang « nạn nhân » để chuyển ra bộ nhớ phụ. Có nhiều thuật toán thay thế trang khác nhau, nhưng tất cả cùng chung một mục tiêu chọn trang « nạn nhân » là trang mà sau khi thay thế sẽ gây ra ít lỗi trang nhất.

Có thể đánh giá hiệu qủa của một thuật toán bằng cách xử lý trên một chuỗi các địa chỉ cần truy xuất và tính toán số lượng lỗi trang phát sinh.

- **FIFO**: chọn các trang đã được nạp vào bộ nhớ trong lâu nhất để thay thế thuật toán này dễ hiểu, dễ cài đặt. Tuy nhiên khi thực hiện không phải lúc nào cũng có kết qủa tốt : trang được chọn để thay thế có thể là trang chứa nhiều dữ liệu cần thiết, thường xuyên được sử dụng nên được nạp sớm, do vậy khi bị chuyển ra bộ nhớ phụ sẽ nhanh chóng gây ra lỗi trang cho những lần truy xuất sau. Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng. Hiện tượng này gọi là nghịch lý Belady.

- **OPT**: thay thế trang sẽ lâu được sử dụng nhất trong tương lai thuật toán này hoàn hảo về mặt ý tưởng nhưng không khả thi về mặt thực tế. Thuật toán này bảo đảm số lượng lỗi trang phát sinh là thấp nhất, nó cũng không gánh chịu nghịch lý Belady, tuy nhiên, đây là một thuật toán không khả thi trong thực tế, vì không thể biết trước chuỗi truy xuất của tiến trình trong tương lai!

- **LRU**: thay thế trang lâu nhất trong bộ nhớ chưa được sử dụng dùng quá khứ gần để đoán tương lai, FIFO để biết thời điểm nạp vào, tối ưu thời điểm sẽ truy cập, LRU đòi hỏi phần cứng phải hỗ trợ khá nhiều: biến bộ đếm, stack.

  - ❖ Giải thuật LRU bất khả thi nhất bởi vì gây ra lỗi trang nhiều, đòi hỏi khá nhiều phần cứng.

Giải thuật phức tạp nhất là OPT vì nó không có thể nhìn thấy hết được trang phát sinh trong tương lai

HẾT